PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-293692
(43)Date of publication of application : 04.11.1998

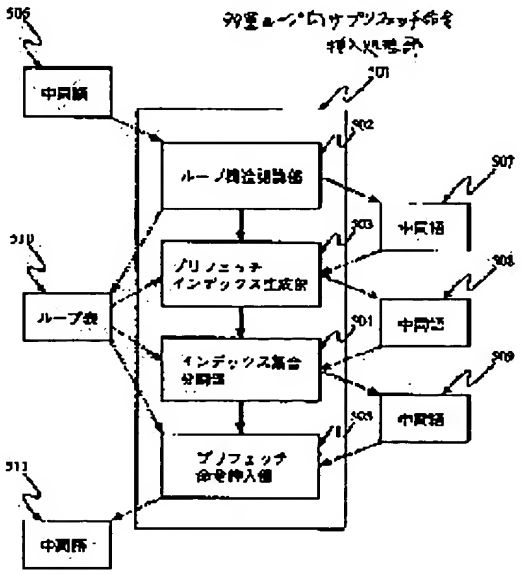
(51)Int.Cl. G06F 9/45
G06F 9/32

(21)Application number : 09-100429
(22)Date of filing : 17.04.1997

(71)Applicant : HITACHI LTD
(72)Inventor : NISHIYAMA HIROYASU

(54) DATA PREFETCH METHOD FOR MULTIPLE LOOP, PROCESSOR AND PROGRAM GENERATING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To shorten an inner-most loop length and to convert a program so as to effectively perform prefetch even to a multiple loop having the long loop length of outside loop.
SOLUTION: The selection of prefetch object loop is performed for the inner- most loop of multiple loop (502). Index cluster division is applied to the selected loop and it is divided into the first half partial loop and the latter half partial loop (504). The prefetch instruction of data to be used for the relevant loop itself is inserted into the first half partial loop, and the prefetch instruction of data to be used for the next prefetch object loop is inserted into the latter half partial loop (505). Thus, the time of wait due to main memory reference can be sufficiently reduced and the execution of computer program can be accelerated.



LEGAL STATUS

[Date of request for examination] 25.08.2000
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 3546341
[Date of registration] 23.04.2004
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－293692

(43)公開日　平成10年(1998)11月４日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	9/45		Ｇ０６Ｆ	9/44	３２２Ｇ
	9/32	３１０		9/32	３１０Ｊ

審査請求　未請求　請求項の数６　ＯＬ　（全 12 頁）

(21)出願番号　特願平9－100429

(22)出願日　平成９年(1997)４月17日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　西山　博泰
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(74)代理人　弁理士　有近　紳志郎

(54)【発明の名称】　多重ループ向けデータプリフェッチ方法、プロセッサおよびプログラム生成方法

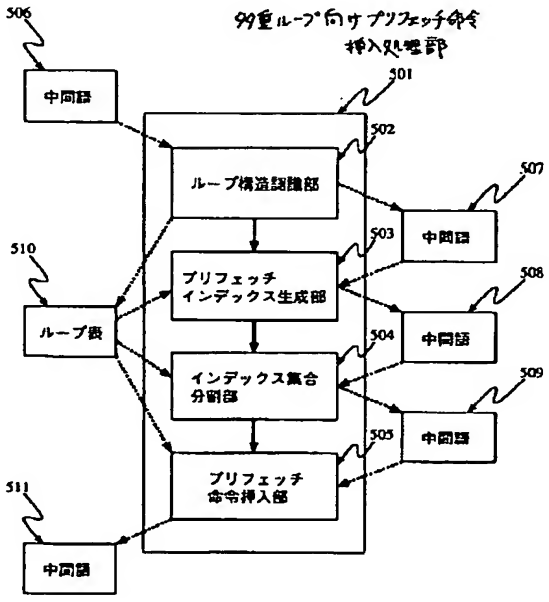
(57)【要約】

【課題】　最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、プリフェッチを有効に行うようにプログラムを変換する。

【解決手段】　多重ループの最内側ループに対して、プリフェッチ対象ループの選択を行なう（５０２）。選択したループに対してインデックス集合分割を適用し、前半部分ループと後半部分ループに分割する（５０４）。前半部分ループには当該ループ自身で使用するデータのプリフェッチ命令を挿入し、後半部分ループには次プリフェッチ対象ループで使用するデータのプリフェッチ命令を挿入する（５０５）。

【効果】　主記憶参照による待ち時間を十分に減少することができ、コンピュータプログラムの実行を高速化できる。

図５



【特許請求の範囲】
【請求項１】　兄弟ループの関係にある２以上の最内側ループをもつ多重ループに対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクル数と最内側ループの予測実行サイクル数とに基づいて分割すべき繰り返し回数を求め、当該最内側ループを、前記分割すべき繰り返し回数に基づく回数のループ繰り返しを実行する前半部分と、残りの繰り返し回数だけループ繰り返しを実行する後半部分とに分割し、当該最内側ループ自身で使用するデータに対するプリフェッチ命令を前記前半部分に挿入し、次に実行する兄弟ループの関係にある最内側ループで使用するデータに対するプリフェッチ命令を前記後半部分に挿入することを特徴とする多重ループ向けデータプリフェッチ方法。
【請求項２】　密多重ループに対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクル数と最内側ループの予測実行サイクル数とに基づいて分割すべき繰り返し回数を求め、当該最内側ループを、前記分割すべき繰り返し回数に基づく回数だけループ繰り返しを実行する前半部分と、残りの繰り返し回数だけループ繰り返しを実行する後半部分とに分割し、当該最内側ループ自身で使用するデータに対するプリフェッチ命令を前記前半部分に挿入し、次の外側ループ繰り返しにおける最内側ループで使用するデータに対するプリフェッチ命令を前記後半部分に挿入することを特徴とする多重ループ向けデータプリフェッチ方法。
【請求項３】　プリフェッチ命令を持つプロセッサにおいて、汎用レジスタとは別に、プリフェッチ命令のベース，オフセットなどのオペランドとして指定することが可能なプリフェッチ用拡張レジスタを備えたことを特徴とするプロセッサ。
【請求項４】　請求項３に記載のプロセッサにおいて、オペランドで指定されたプリフェッチ用拡張レジスタの値をプリフェッチ対象アドレスの計算のためのベースあるいはオフセットとし、別オペランドで指定された汎用レジスタの値と組み合わせてプリフェッチ対象アドレスを計算するプリフェッチ命令を備えたことを特徴とするプロセッサ。
【請求項５】　請求項３または請求項４に記載のプロセッサにおいて、プリフェッチ用拡張レジスタと、汎用レジスタあるいは主記憶との間でデータの転送を行なうデータ転送命令を備えたことを特徴とするプロセッサ。
【請求項６】　与えられた入力プログラムを解析し、請求項１または請求項２に記載の多重ループ向けデータプリフェッチ方法を適用し、プリフェッチ命令を挿入した出力プログラムを生成することを特徴とするプログラム生成方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、多重ループ向けデータプリフェッチ方法、プロセッサおよびプログラム生成方法に関し、さらに詳しくは、最内側ループ長が短く、外側ループのループ長が長い多重ループに対しても、主記憶参照による待ち時間を十分に減少することが出来る多重ループ向けデータプリフェッチ方法、プロセッサおよびプログラム生成方法に関する。
【０００２】
【従来の技術】コンピュータでは、主記憶よりも高速なキャッシュメモリをプロセッサと主記憶の間に配置し、最近参照したデータをキャッシュメモリ上に置くことによって、主記憶参照による待ち時間を減少させている。ところが、数値計算処理など大規模データを使用する計算では、データの参照局所性が低いためキャッシュミスが多発し、主記憶参照による待ち時間を十分に減少することが出来ない問題点があった。
【０００３】このような大規模データに対するキャッシュミスに対処するため、例えば、文献「T.C.Mowry 他, Design and Evaluation of a Compiler Algorithmfor Prefetching, Proceedings of the 5th International Conference on ArchitecturalSupport for Programming Languages and Operating Systems,pp.62-73,1992」に示されているように、データを使用する時より先行して主記憶からキャッシュメモリへデータを移動するプリフェッチ命令をプロセッサに用意し、コンパイラによってプログラム中にプリフェッチ命令を挿入するプリフェッチ方法が提案されている。具体的には、図１３の(a)に示すようなループ２０１に対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクル数とループの予測実行サイクル数に基づいてデータをプリフェッチすべき要素間のオフセットαを計算し、まず、図１３の(b)のループ２０２のように、データを使用するループよりもオフセットαだけ先行したループでデータをプリフェッチするようにプリフェッチ命令“PREFETCH”を挿入する。しかし、これだけでは、１～α回の繰り返しで使用するデータはプリフェッチされない。また、最後の（Ｎ－α＋１）～Ｎ回の繰り返しでは、演算に使用されないデータをプリフェッチすることになる。そこで、次に、図１３の(c)に示すように、１～α回の繰り返しで使用するデータのプリフェッチだけを行なうα回のループ２０３をループ開始前に挿入する。また、インデックス集合分割を適用して元のループ２０１を１～（Ｎ－α）回の繰り返しを実行する前半部分ループ２０４と残りの繰り返しを実行する後半部分ループ２０５に分割し、後半部分ループ２０５にはプリフェッチ命令を挿入しないようにする。
【０００４】
【発明が解決しようとする課題】上記のようなプリフェッチ方法によれば、キャッシュミスが減り、主記憶参照による待ち時間を減少することが出来る。ところで、プリフェッチの本質は主記憶からキャッシュメモリへのデ

ータの移動と演算がオーバラップして行なわれる図１３のループ２０４にあるが、ループ長Ｎに対してオフセットαの値が比較的大きな場合には、この本質的なループ２０４の比率が小さくなり、本質的でないループ２０３，２０５の比率が大きくなってしまう。このため、全体として主記憶参照による待ち時間を十分に減少することが出来なくなる問題点がある。すなわち、従来は最内側ループのみをプリフェッチ方法の適用対象としていたため、最内側ループ長が短く、外側ループのループ長が長い多重ループに対しては、主記憶参照による待ち時間を十分に減少することが出来ない問題点があった。そこで、本発明の目的は、最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、主記憶参照による待ち時間を十分に減少することが出来る多重ループ向けデータプリフェッチ方法、マイクロプロセッサおよびプログラム生成方法を提供することにある。

【０００５】

【課題を解決するための手段】本発明では、次に示す手順によりプリフェッチ命令を挿入する。

（手順１）．ディレクティブやオプションによるユーザからの指定や、上記論文に示されているようなデータの再利用性を考慮したコンパイラの解析によって、多重ループの最内側ループの中からプリフェッチを適用するループLOOPo，…，LOOPm-1を選択する。

（手順２）．ループLOOPｉ（０≦ｉ≦ｍ－１）の繰り返し実行回数をLength(ｉ)とし、主記憶からキャッシュメモリへのデータ転送に要するサイクル数をCycle(MEM)とし、ループLOOPｉの１回の繰り返しの予測実行サイクル数をCycle(ｉ)とするとき、分割すべき繰り返し回数αをα＝Cycle(MEM)／Cycle(ｉ)により求める。そして、ループLOOPｉに対してインデックス集合分割を適用し、１～（Length(ｉ)－α）回の繰り返しを実行する前半部分ループLOOPｉ．０と、残りの（Length(ｉ)－α＋１）～Length(ｉ)回の繰り返しを実行する後半部分ループLOOPｉ．１とに分割する。

（手順３）．前半部分ループLOOPｉ．０でキャッシュミスを生じるメモリ参照“X[j]”に対して、プリフェッチ命令“PREFETCH X[j+Step(ｉ)＊α]”を挿入する。ここで、Step（ｉ）は、LOOPｉのループインデックスの増分値である。また、後半部分ループLOOPｉ．１のループインデックスをｋとし、その初期値をInit(i.1)とするとき、ループLOOPｉの次のプリフェッチ対象ループLOOP（(i+1) mod m）でキャッシュミスを生じるメモリ参照“Y[j]”に対して、その初期参照インデックスをStart(Y)とすると、プリフェッチ命令“PREFETCH Y[Start(Y)+(k-Init(i.1))＊(Step(LOOP((i+1) mod m))／Step(ｉ))]”を挿入する。但し、（(i+1) mod m）＝０ならば、外側ループインデックスを１回分進めたアドレスをプリフェッチ対象アドレスとする。なお、（Ａ mod Ｂ）は、ＡをＢで割った余りを表すものとする。

【０００６】さて、第１の観点では、本発明は、兄弟ループの関係にある２以上の最内側ループをもつ多重ループに対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクルと最内側ループの予測実行サイクルとに基づいて分割すべき繰り返し回数を求め、当該最内側ループを、前記分割すべき繰り返し回数に基づく回数の繰り返しを実行する前半部分と、残りの繰り返し回数だけ繰り返しを実行する後半部分とに分割し、当該最内側ループ自身で使用するデータに対するプリフェッチ命令を前記前半部分に挿入し、次に実行する兄弟ループの関係にある最内側ループで使用するデータに対するプリフェッチ命令を前記後半部分に挿入することを特徴とする多重ループ向けデータプリフェッチ方法を提供する。上記第１の観点による多重ループ向けデータプリフェッチ方法は、先述の（手順３）で、（(i+1) mod m）≠０ の場合に相当し、後半部分ループにおいても、主記憶からキャッシュメモリへのデータ（＝次に実行する兄弟ループの関係にある最内側ループで使用するデータ）の移動と演算がオーバラップして行なわれることとなり、プリフェッチの本質的な作用により、最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、主記憶参照による待ち時間を十分に減少することが出来る。

【０００７】第２の観点では、本発明は、密多重ループに対して、主記憶からキャッシュメモリへのプリフェッチに要するサイクルと最内側ループの予測実行サイクルとに基づいて分割すべき繰り返し回数を求め、当該最内側ループを、前記分割すべき繰り返し回数に基づく回数だけループ繰り返しを実行する前半部分と、残りの繰り返し回数だけループ繰り返しを実行する後半部分とに分割し、当該最内側ループ自身で使用するデータに対するプリフェッチ命令を前記前半部分に挿入し、次の外側ループ繰り返しにおける最内側ループで使用するデータに対するプリフェッチ命令を前記後半部分に挿入することを特徴とする多重ループ向けデータプリフェッチ方法を提供する。上記第２の観点による多重ループ向けデータプリフェッチ方法は、先述の（手順３）で、（(i+1) mod m）＝０ の場合に相当し、後半部分ループにおいても、主記憶からキャッシュメモリへのデータ（＝次の外側ループ繰り返しにおける最内側ループで使用するデータ）の移動と演算がオーバラップして行なわれることとなり、プリフェッチの本質的な作用により、最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、主記憶参照による待ち時間を十分に減少することが出来る。

【０００８】第３の観点では、本発明は、プリフェッチ命令を持つプロセッサにおいて、汎用レジスタとは別に、プリフェッチ命令のベース，オフセットなどのオペランドとして指定することが可能なプリフェッチ用拡張レジスタを備えたことを特徴とするプロセッサを提供す

る。プリフェッチでは、プリフェッチのオフセットやプリフェッチ対象アドレスを生成するのにレジスタを使用する。このため、多数の汎用レジスタを使用するプログラムではレジスタ不足を生じ、メモリへのスピル処理により、性能が低下するおそれがある。そこで、上記第３の観点によるプロセッサでは、プリフェッチ用拡張レジスタを備えた。これにより、上記第１の観点または第２の観点による多重ループ向けデータプリフェッチ方法を実施する際にレジスタ不足が生じることを防止できる。なお、上記プリフェッチ用拡張レジスタは、その参照に要するサイクル数がメモリ参照の場合と異なり常に一定であり、プリフェッチだけでなく、汎用レジスタ不足時の一時的なデータ保存のためにも使用できる。

【０００９】第４の観点では、本発明は、上記構成のプロセッサにおいて、オペランドで指定されたプリフェッチ用拡張レジスタの値をプリフェッチ対象アドレスの計算のためのベースあるいはオフセットとし、別オペランドで指定された汎用レジスタの値と組み合わせてプリフェッチ対象アドレスを計算するプリフェッチ命令を備えたことを特徴とするプロセッサを提供する。上記第４の観点によるプロセッサでは、プリフェッチ対象アドレスを計算するのに使用するプリフェッチ用拡張レジスタと汎用レジスタをオペランドで指定できるプリフェッチ命令を備えたため、上記第１の観点または第２の観点による多重ループ向けデータプリフェッチ方法を効率よく実行できる。

【００１０】第５の観点では、本発明は、上記構成のプロセッサにおいて、プリフェッチ用拡張レジスタと、汎用レジスタあるいは主記憶との間でデータの転送を行なうデータ転送命令を備えたことを特徴とするプロセッサを提供する。上記第５の観点によるプロセッサでは、プリフェッチ用拡張レジスタに値を設定したり、プリフェッチ用拡張レジスタの値を取り出したりすることを、上記データ転送命令により実行できる。このため、上記第１の観点または第２の観点による多重ループ向けデータプリフェッチ方法を効率よく実行できる。

【００１１】第６の観点では、本発明は、与えられた入力プログラムを解析し、請求項１または請求項２に記載の多重ループ向けデータプリフェッチ方法を適用し、プリフェッチ命令を挿入した出力プログラムを生成することを特徴とするプログラム生成方法を提供する。上記第６の観点によるプログラム生成方法をコンピュータに実施させるコンパイラ・プログラムを記録した記録媒体をコンピュータに読み取らせて、入力プログラムに対して上記第１の観点または第２の観点による多重ループ向けデータプリフェッチ方法を適用することをコンピュータに実施させれば、実行効率の高い出力プログラムを得ることが出来る。

【００１２】

【発明の実施の形態】以下、図面を参照しながら、本発明の実施形態を説明する。なお、これにより本発明が限定されるものではない。

【００１３】図１は、本発明を実施するコンピュータシステムの一例である。このコンピュータシステム１００は、マイクロプロセッサ１０１と、そのマイクロプロセッサ１０１に内蔵されたキャッシュメモリ１０２と、主記憶１０３と、ディスク装置１０４とを具備してなる。磁気ディスク装置１０４には、記録媒体から読み込まれたコンパイラ・プログラムが格納されている。マイクロプロセッサ１０１は、ディスク装置１０４に格納されたコンパイラ・プログラムを読み出し、与えられたソースプログラムに対してコンパイル処理を行い、コンパイル結果のオブジェクトプログラムを出力する。

【００１４】マイクロプロセッサ１０１は、通常のメモリ参照命令の実行を行なう場合には、まずキャッシュメモリ１０２に参照対象データがあるかどうかを調べ、キャッシュメモリ１０２に当該データが存在すればそのデータを参照し、キャッシュメモリ１０２に参照対象データが存在しなければ主記憶１０３上の当該データを参照すると共に当該データの属するキャッシュブロックをキャッシュメモリ１０２にコピーする。もし、主記憶１０３からキャッシュメモリ１０２へキャッシュブロックを移動するのに十分なサイクル数だけ前に参照対象データに対するプリフェッチ命令が発行されておれば、参照対象データが必ずキャッシュメモリ１０２に存在し、当該データを主記憶１０３から参照するための待ち時間が無くなり、プログラムの実行性能が向上する。

【００１５】図２は、従来のプリフェッチ方法（ａ）と本発明の多重ループ向けプリフェッチ方法（ｂ）とを比較した説明図である。これについては、後で詳述する。

【００１６】図３は、兄弟ループの関係にある２つの最内側ループ３０１，３０２をもつ多重ループに対して、本発明の多重ループ向けプリフェッチ方法を適用した結果の説明図である。これについては、後で詳述する。

【００１７】図４は、密多重ループに対して、本発明の多重ループ向けプリフェッチ方法を適用した結果の説明図である。これについては、後で詳述する。

【００１８】図５は、コンパイラ・プログラムの要部である多重ループ向けプリフェッチ命令挿入処理部５０１の構成図である。なお、実線の矢印は制御の流れを表し、破線の矢印はデータの流れを表している。この多重ループ向けプリフェッチ命令挿入処理部５０１は、本発明の多重ループ向けプリフェッチ方法を実施する要部であり、ソースプログラムから変換した中間語５０６を入力とし、多重ループ向けプリフェッチを行なうように変換した中間語５１１を出力とする。多重ループ向けプリフェッチ命令挿入処理部５０１は、ループ構造認識部５０２と、プリフェッチインデックス生成部５０３と、インデックス集合分割部５０４と、プリフェッチ命令挿入部５０５とを具備してなる。

【００１９】図６は、前記ループ構造認識部５０２の処理動作を示すフローチャートである。処理６０１では、処理を開始する。処理６０２では、Ｌ１に最内側ループの外側ループ集合を求める。ループ集合を求める処理は、例えば「Aho他、Compilers - Principles, Techniques, and Tools, Addison-Wesley, 1986」に述べられているような既知の制御フロー解析技術を適用することにより実現できる。処理６０３では、求めたループ集合Ｌ１が空集合か否かをチェックし、空集合であれば処理６１０へ制御を移し、処理を終了する。Ｌ１が空集合でなければ、処理６０４へ制御を移す。処理６０４では、Ｌ１より要素を１つ取り出し、１１に格納する。また、１１の子ループ集合をＬ０に格納する。さらに、Ｌ１のプリフェッチ対象ループを求める集合Ｓを空集合に初期化する。処理６０５では、１１の子ループ集合Ｌ０が空集合かどうか確かめ、空集合であれば処理６０９へ進み、空集合でなければ処理６０６へ制御を移す。処理６０６では、子ループ集合Ｌ０より要素を１つ取り出し、そのループを１０とする。処理６０７では、ループ１０にプリフェッチを適用するか否かを判定する。この判定は、ユーザからのオプションやディレクティブによる指定や、前記T.C.Mowry 他による論文に示されている既知技術によって行えばよい。そして、プリフェッチを適用する場合は処理６０８へ制御を移し、適用しない場合は前記処理６０５に制御を移し、次の最内側ループを処理する。処理６０８では、ループ１０をプリフェッチ対象ループとして集合Ｓに追加し、前記処理６０５へ制御を移し、次の最内側ループの処理を行なう。処理６０９では、ループ表の１１の欄にプリフェッチ対象ループの集合Ｓを登録し、前記処理６０３に制御を移し、次の多重ループを処理する。

【００２０】図７は、前記プリフェッチインデックス生成部５０３の処理動作を示すフローチャートである。処理７０１では、処理を開始する。処理７０２では、集合Ｌ１に最内側ループの外側ループ集合を求める。処理７０３では、求めたループ集合Ｌ１が空集合か否かをチェックし、空集合であれば処理７０７へ制御を移し、処理を終了する。Ｌ１が空集合でなければ、処理７０４へ制御を移す。処理７０４では、Ｌ１より要素を１つ取り出し、１１に格納する。また、ループ構造認識部５０２が求めたプリフェッチ対象ループ集合１１を１０に格納する。また、変数ｉを“１”に初期化する。さらに、変数Ｑを１０の要素数に初期化する。処理７０５では、変数ｉがＱ以上になったか否かを判定し、なった場合は前記処理７０３に制御を移し、次の多重ループの処理を行なう。変数ｉがＱより小さい場合は処理７０６へ制御を移す。処理７０６では、１０にＬ０のｉ番目のループを格納する。また、１０’にＬ０の（（ｉ mod Q)＋１）番目のループを格納する。ここで、(ｉ mod Q)は、ｉをＱで割った余りを表す。また、１０’で参照するデータの初期アドレスの計算式を１０の前に生成する。さらに、ｉの値を“１”増加し、前記処理７０５に制御を移して、次の最内側ループの処理を行なう。

【００２１】図８は、前記インデックス集合分割部５０４の処理動作を示すフローチャートである。処理８０１では、処理を開始する。処理８０２では、集合Ｌ１に最内側ループの外側ループ集合を求める。処理８０３では、求めたループ集合Ｌ１が空集合か否かをチェックし、空集合であれば処理８０７へ制御を移し、処理を終了する。Ｌ１が空集合でなければ、処理８０４へ制御を移す。処理８０４では、Ｌ１より要素を１つ取り出し、１１に格納する。また、ループ構造認識部５０２が求めたプリフェッチ対象ループ集合１１を１０に格納する。また、変数ｉを“１”に初期化する。さらに、変数Ｑを１０の要素数に初期化する。処理８０５では、変数ｉがＱ以上になったか否かを判定し、なった場合は前記処理８０３に制御を移し、次の多重ループの処理を行なう。変数ｉがＱより小さい場合は処理８０６へ制御を移す。処理８０６では、１０にＬ０のｉ番目のループを格納する。また、αに主記憶１０３からキャッシュメモリ１０２へのプリフェッチに要するサイクル数をループ１０の１回当たりの予測実行サイクル数で割った値、すなわち、分割すべき繰り返し回数を格納する。また、ループ１０にインデックス集合分割を適用して、１～（Ｎ－α）回の繰り返しを実行するループと（Ｎ－α＋１）～Ｎ回の繰り返しを実行するループとに分割する。このインデックス集合分割は、例えば「M.Wolfe、High Performance Compilers For Parallel Computing、Addison-Wesley,1996」に述べられている既知のループ最適化技法を適用すればよい。さらに、ｉの値を“１”増加し、前記処理８０５に制御を移して、次の最内側ループの処理を行なう。

【００２２】図９は、前記プリフェッチ命令挿入部５０５の処理動作を示すフローチャートである。処理９０１では、処理を開始する。処理９０２では、集合Ｌ１に最内側ループの外側ループ集合を求める。処理９０３では、求めたループ集合Ｌ１が空集合か否かをチェックし、空集合であれば処理９０７へ制御を移し、処理を終了する。Ｌ１が空集合でなければ、処理９０４へ制御を移す。処理９０４では、Ｌ１より要素を１つ取り出し、１１に格納する。また、ループ構造認識部５０２が求めたプリフェッチ対象ループ集合１１を１０に格納する。また、変数ｉを“１”に初期化する。さらに、変数Ｑを１０の要素数に初期化する。処理９０５では、変数ｉがＱ以上になったか否かを判定し、なった場合は前記処理９０３に制御を移し、次の多重ループの処理を行なう。変数ｉがＱより小さい場合は処理９０６へ制御を移す。処理９０６では、１０にＬ０のｉ番目のループを格納する。また、１０’にＬ０の（（ｉ mod Q)＋１）番目のループを格納する。また、１０．０にインデックス集合分

割部５０４が１０をインデックス集合分割してできた前半部分ループを格納し、１０．１に後半部分ループを格納する。また、αに主記憶１０３からキャッシュメモリ１０２へのプリフェッチに要するサイクル数をループ１０の１回当たりの予測実行サイクル数で割った値、すなわち、分割すべき繰り返し回数を格納する。また、ループ１０のプリフェッチ対象メモリ参照Ｘ［ｉ］に対して、インデックス集合分割によって生成した前半部分ループ１０．０にＸ［ｉ＋Step｛ｉ｝＊α］に対するプリフェッチ命令を挿入する。また、１０’のプリフェッチ対象メモリ参照Ｙ［ｊ］に対して、１０．１のループインデックスをｋとし、βを１０．１開始時のループインデックスの値とし、Ｄをループ１０とループ１０’のループ増分値の比とすると、後半部分ループ１０．１にＹ［Start(Y)＋（ｋ－β）Ｄ］に対するプリフェッチ命令を挿入する。さらに、ｉの値を“１”増加し、前記処理９０５に制御を移して、次の最内側ループの処理を行なう。

【００２３】プリフェッチ命令のアドレス計算式中のαやβなどはループ中で不変な値であり、ループ外へ移動することでループ中の演算数を削減することができる。ただし、これらの式をループ外へ移動すると、その値を保持するための汎用レジスタが別途必要となるため、汎用レジスタ使用数の多いプログラムでは汎用レジスタ不足を生じるおそれがある。そこで、マイクロプロセッサ１０１にプリフェッチ用拡張レジスタを設け、そのプリフェッチ用拡張レジスタに前記αやβの値を格納し、このプリフェッチ用拡張レジスタを使用するプリフェッチ命令を生成すれば、汎用レジスタの不足を生じさせずに済む。図１０に、プリフェッチ用拡張レジスタを使用するプリフェッチ命令の一例を示す。ＧＲｎは、汎用レジスタを表している。また、ＰＦＲｎは、プリフェッチ用拡張レジスタを表している。このプリフェッチ命令を使用し、ループ実行中に不変なオフセットについてはプリフェッチ用拡張レジスタＰＦＲｎを指定し、ループ実行時に可変なベース値については汎用レジスタＧＲｎを指定するようにすれば、プログラム実行に必要な汎用レジスタ数を増すことなく、プリフェッチを行なうプログラムを生成することができる。図１１は、汎用レジスタＧＲｎに格納された値をプリフェッチ用拡張レジスタＰＦＲｎへ設定する命令の一例である。また、図１２は、プリフェッチ用拡張レジスタＰＦＲｎより汎用レジスタＧＲｎへ値をコピーする命令である。

【００２４】次に、本発明のプリフェッチ方法の適用例を説明する。図３は、兄弟ループの関係にある２つの最内側ループ３０１，３０２をもつ多重ループに対して本発明のプリフェッチ方法を適用した例である。図３の(ａ)はプリフェッチ命令挿入前のプログラムであり、図３の(ｂ)はプリフェッチ命令挿入後のプログラムである。次のステップ１～ステップ５により、図３の(ａ)のプログラムから図３の(ｂ)のプログラムが得られる。

ステップ１：プリフェッチ対象の最内側ループとしてループ３０１とループ３０２を選択する。

ステップ２：プリフェッチ対象の最内側ループ３０１と３０２に対して、インデックス集合分割を適用する。

ステップ３：ループ３０１を分割して生成した前半部分ループに、ループ３０１に対するプリフェッチ命令を挿入する。同様に、ループ３０２を分割して生成した前半部分ループに、ループ３０２に対するプリフェッチ命令を挿入する。

ステップ４：ループ３０１を分割して生成した後半部分ループに、ループ３０２に対するプリフェッチ命令を挿入する。同様に、ループ３０２を分割して生成した後半部分ループに、外側ループの次回の繰り返しでのループ３０１に対するプリフェッチ命令を挿入する。

ステップ５：外側ループの直前に、外側ループの最初の繰り返しでループ３０１が最初のα回のループ繰り返しで参照するデータのプリフェッチを挿入する。

【００２５】ループ３０３は、最初の外側ループ繰り返しで元のループ３０１の１～α回の演算で使用するデータのプリフェッチを行なうループであり、上記ステップ５で挿入される。ループ３０４は、元のループ３０１の１～（Ｎ－α）回の演算と同時に（α＋１）～Ｎ回の演算で使用するデータをプリフェッチするループであり、上記ステップ３で挿入される。ループ３０５は、元のループ３０１の（Ｎ－α＋１）～Ｎ回の演算と同時に、元のループ３０２の最初の１～α回の演算で使用するデータをプリフェッチするループであり、上記ステップ４で挿入される。ループ３０６は、元のループ３０１のループ長がαより短い場合にループ３０２で使用するデータのプリフェッチとループ３０１の演算を同時に行なうループであり、上記ステップ４で挿入される（この場合、前半部分ループがなく、後半部分ループだけがある）。ループ３０７は、元のループ３０２の最初の１～（Ｎ－α）回の演算と同時に（α＋１）～Ｎ回の実行で使用するデータをプリフェッチするループであり、上記ステップ３で挿入される。ループ３０８は、元のループ３０２の（Ｎ－α＋１）～Ｎ回の演算の実行と同時に外側ループの次回の繰り返しでのループ３０１の１～α回の演算で使用するデータをプリフェッチするループであり、上記ステップ４で挿入される。ループ３０９は、元のループ３０２のループ長がαより短い場合にループ３０１で使用するデータのプリフェッチとループ３０２の演算を同時に行なうループであり、上記ステップ４で挿入される（この場合、前半部分ループがなく、後半部分ループだけがある）。

【００２６】図４は、多重ループの特殊な形態である密多重ループに本発明のプリフェッチ方法を適用した例である。図４の(ａ)はプリフェッチ命令挿入前のプログラムであり、図４の(ｂ)はプリフェッチ命令挿入後のプロ

グラムである。密多重ループの場合、最内側には1つのループしかないため、最内側ループの実行と次外側ループ繰り返しの最内側ループで使用するデータのプリフェッチを同時に行なう。ループ402は、最初の外側ループ繰り返しで元のループ401の最初の1～α回の演算で使用するデータのプリフェッチを行なうループである。ループ403は、元のループ401の1～（N－α）回の演算と同時に（α＋1）～N回の演算で使用するデータをプリフェッチするループである。ループ404は、元のループ401の（N－α＋1）～N回の演算と同時に次外側ループ繰り返しのループ401の最初の1～α回の演算で使用するデータのプリフェッチを行なうループである。ループ405は、ループ401のループ長がαより短い場合にループ401の演算と同時に次外側ループ繰り返しでループ401が使用するデータのプリフェッチを行なうループである。

【0027】以上の結果、図2の(a)に示すように、従来のプリフェッチ方法では次外側ループ開始時にプリフェッチだけを行い演算を行わない空きサイクルを生じていたのに対して、図2の(b)に示すように、本発明の多重ループ向けデータプリフェッチ方法では、インデックス集合分割を適用してできた後半部分ループで次外側ループ繰り返しで参照するデータのプリフェッチを行なっているため、空きサイクルを生じない（開始時点を除く）。

【0028】

【発明の効果】本発明の多重ループ向けデータプリフェッチ方法、プロセッサおよびプログラム生成方法によれば、最内側ループ長が短かく、外側ループのループ長が長い多重ループに対しても、プリフェッチを有効に行うようにプログラムを変換できるため、主記憶参照による待ち時間を十分に減少することができ、これによりコンピュータプログラムの実行を高速化できる。

【図面の簡単な説明】

【図1】本発明の多重ループ向けデータプリフェッチ方法を実施するコンピュータシステムの構成図である。

【図2】従来のプリフェッチ方法と本発明の多重ループ向けデータプリフェッチ方法の比較説明図である。

【図3】兄弟ループの関係にある複数の最内側ループをもつ多重ループに対して本発明のプリフェッチ方法を適用した例の説明図である。

【図4】密多重ループに対して本発明のプリフェッチ方法を適用した例の説明図である。

【図5】コンパイラ・プログラムの要部である多重ループ向けプリフェッチ命令挿入処理部501の構成図である。

【図6】ループ構造認識部の処理動作を示すフローチャートである。

【図7】プリフェッチインデックス生成部の処理動作を示すフローチャートである。

【図8】インデックス集合分割部の処理動作を示すフローチャートである。

【図9】プリフェッチ命令挿入部の処理動作を示すフローチャートである。

【図10】プリフェッチ命令の説明図である。

【図11】プリフェッチ用拡張レジスタへのデータ転送命令の説明図である。

【図12】プリフェッチ用拡張レジスタからのデータ転送命令の説明図である。

【図13】従来のプリフェッチ方法の適用例の説明図である。

【符号の説明】

100	コンピュータシステム
101	マイクロプロセッサ
102	キャッシュメモリ
103	主記憶
104	ディスク装置
301，302	兄弟ループの関係にある最内側ループ
304，307	前半部分ループ
305，306，308，309	後半部分ループ
401	密多重ループ
403	前半部分ループ
404，405	後半部分ループ
501	多重ループ向けプリフェッチ命令挿入処理部
502	ループ構造認識部
503	プリフェッチインデックス生成部
504	インデックス集合分割部
505	プリフェッチ命令挿入部

【図10】

図10

```
prefetch GRn, PFRn
```

【図11】

図11

```
move_to_pfr PFRn, GRn
```

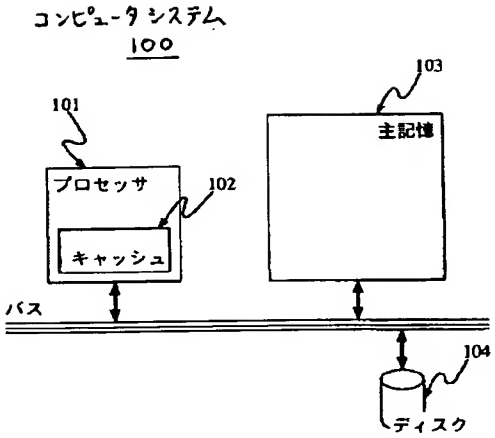
【図12】

図12

```
move_from_pfr GRn, PFRn
```

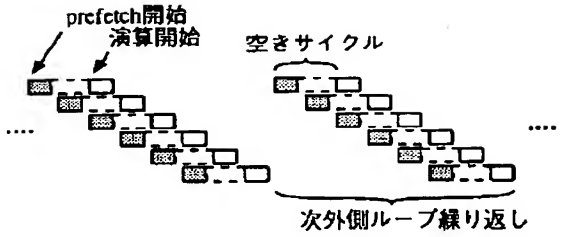
【図１】

図１

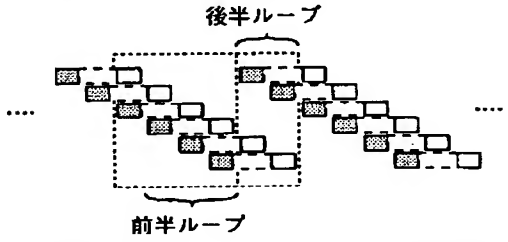


【図２】

図２



(a)　従来方式によりプリフェッチを行なった場合



(b)　本発明の方式によりプリフェッチを行なった場合

【図４】

図４

```
DO J=1,M                          ┐
  DO I=1,N                        │
    X[J][I] = Y[J][I]             ├ 401
  END DO                          │
END DO                            ┘
```

(a)元の密多重ループ

```
DO I=1,α                          ┐
  PREFETCH X[1][I]                │
  PREFETCH Y[1][I]                ├ 402
END DO                            ┘
DO J=1,M
  IF N > α THEN
    DO I=1,N-α                    ┐
      PREFETCH X[J][I+α]          │
      PREFETCH Y[J][I+α]          ├ 403
      X[J][I] = Y[J][I]           │
    END DO                        ┘
    DO I=N-α+1,N                  ┐
      PREFETCH X[J+1][I-N+α]      │
      PREFETCH Y[J+1][I-N+α]      ├ 404
      X[J][I] = Y[J][I]           │
    END DO                        ┘
  ELSE
    DO I=1,N                      ┐
      PREFETCH X[J+1][I]          │
      PREFETCH Y[J+1][I]          ├ 405
      X[J][I] = Y[J][I]           │
    END DO                        ┘
  END IF
END DO
```

(b)多重ループ対するプリフェッチを
行なうよう変換した密多重ループ

【図３】

図３

```
DO J=1,M
  DO I=1,N                    } 301
    V[J][I] = W[J][I]
  END DO
  DO I=1,N                    } 302
    X[J][I] = Y[J][I]
  END DO
END DO
```

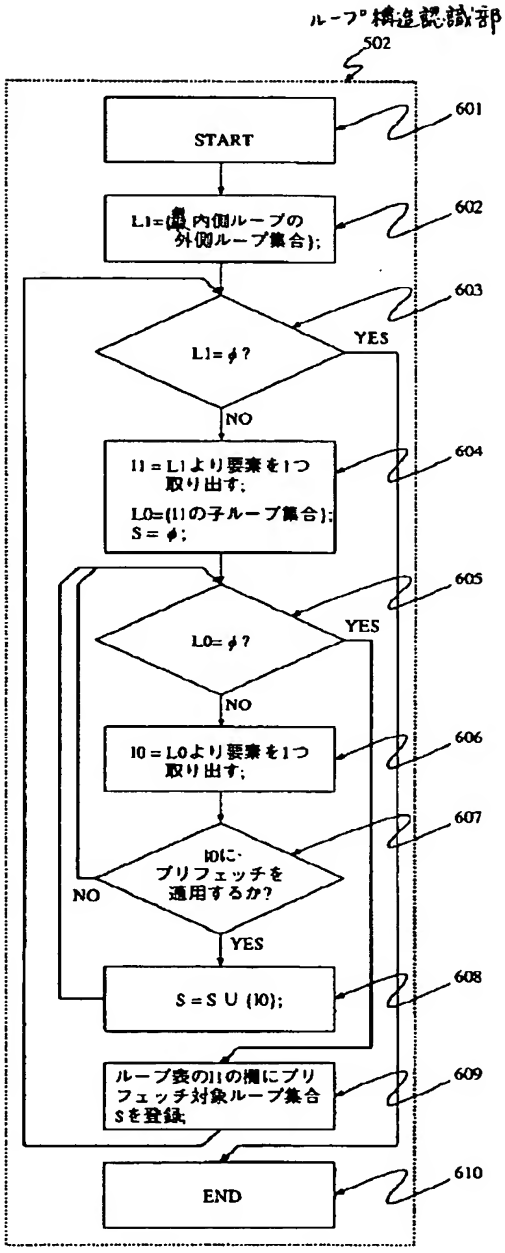
(a)元の　多重ループ

```
DO I=1,α                      } 303
  PREFETCH V[1][I]
  PREFETCH W[1][I]
END DO
DO J=1,M
  IF N>α THEN
    DO I=1,N-α
      PREFETCH V[J][I+α]      } 304
      PREFETCH W[J][I+α]
      V[J][I] = W[J][I]
    END DO
    DO I=N-α+1,N
      PREFETCH X[J][I-N+α]    } 305
      PREFETCH Y[J][I-N+α]
      V[J][I] = W[J][I]
    END DO
  ELSE
    DO I=1,N
      PREFETCH X[J][I-N+α]    } 306
      PREFETCH Y[J][I-N+α]
      V[J][I] = W[J][I]
    END DO
  END IF
  IF N>α THEN
    DO I=1,N-α
      PREFETCH X[J][I+α]      } 307
      PREFETCH Y[J][I+α]
      X[J][I] = Y[J][I]
    END DO
    DO I=N-α+1,N
      PREFETCH V[J+1][I-N+α]  } 308
      PREFETCH W[J+1][I-N+α]
      X[J][I] = Y[J][I]
    END DO
  ELSE
    DO I=1,N
      PREFETCH V[J+1][I-N+α]  } 309
      PREFETCH W[J+1][I-N+α]
      X[J][I] = Y[J][I]
    END DO
  END IF
END DO
```

(b)多重ループに対するプリフェッチを行なう場合のループ

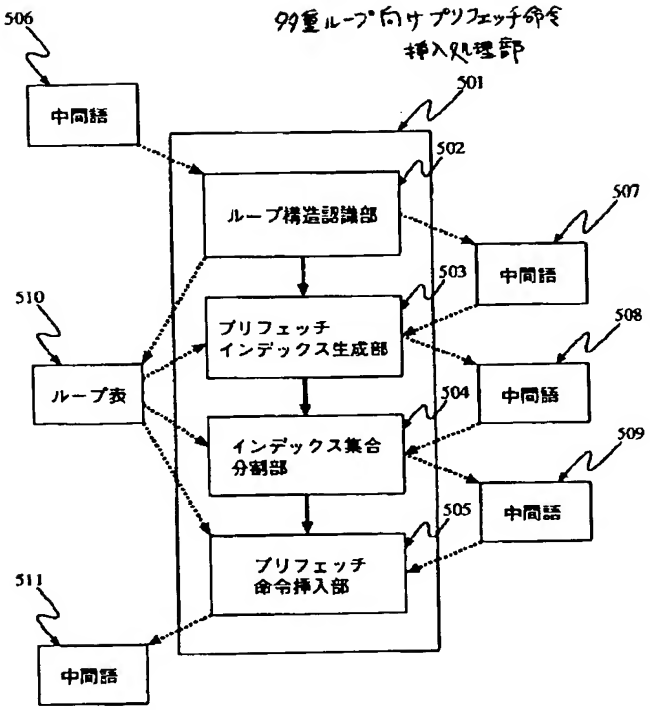
【図６】

図６



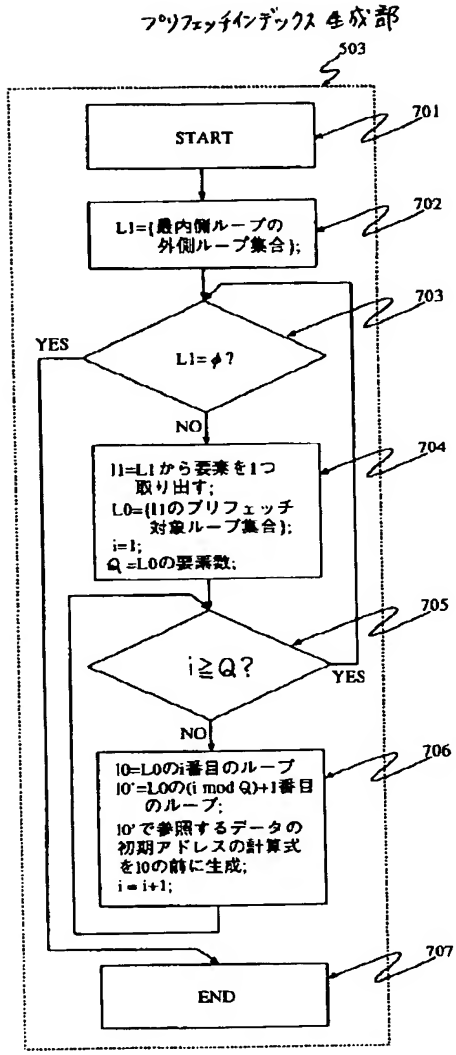
【図５】

図５


多重ループ向けプリフェッチ命令挿入処理部
506
中間語
501
502
ループ構造認識部
507
中間語
503
プリフェッチインデックス生成部
510
ループ表
508
中間語
504
インデックス集合分割部
509
中間語
505
プリフェッチ命令挿入部
511
中間語

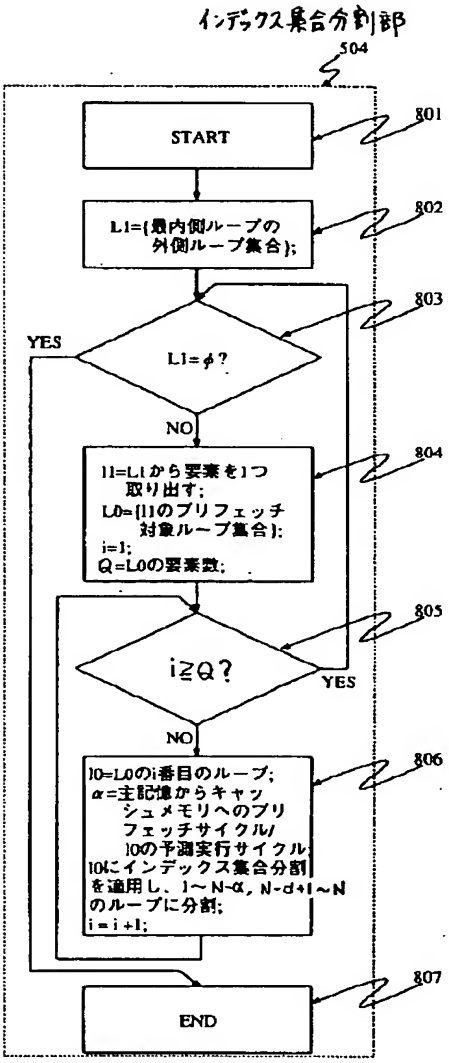
【図７】

図７


プリフェッチインデックス生成部
503
START
701
L1={最内側ループの外側ループ集合};
702
703
L1=φ?
YES
NO
l1=L1から要素を1つ取り出す;
L0={l1のプリフェッチ対象ループ集合};
i=1;
Q=L0の要素数;
704
705
i≧Q?
YES
NO
l0=L0のi番目のループ
l0'=L0の(i mod Q)+1番目のループ;
l0'で参照するデータの初期アドレスの計算式をl0の前に生成;
i = i+1;
706
END
707

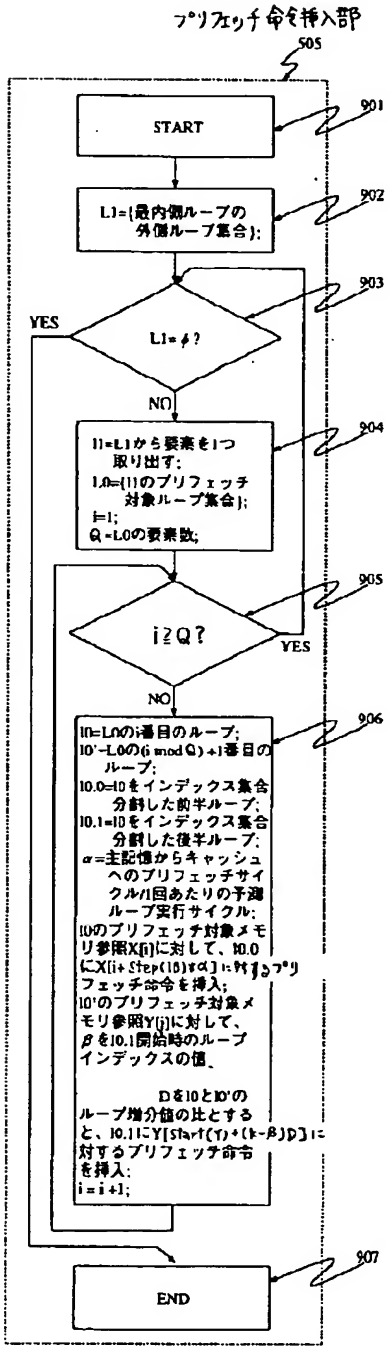
【図8】

図8


インデックス集合分割部
504
START
801
L1={最内側ループの外側ループ集合};
802
L1=φ?
803
YES
NO
l1=L1から要素を1つ取り出す;
L0={l1のプリフェッチ対象ループ集合};
i=1;
Q=L0の要素数;
804
i≧Q?
805
YES
NO
l0=L0のi番目のループ;
α=主記憶からキャッシュメモリへのプリフェッチサイクル/l0の予測実行サイクル;
l0にインデックス集合分割を適用し、1～N-α, N-d+1～Nのループに分割;
i=i+1;
806
END
807

【図9】

図9


プリフェッチ命令挿入部
505
START
901
L1={最内側ループの外側ループ集合};
902
L1=φ?
903
YES
NO
l1=L1から要素を1つ取り出す;
L0={l1のプリフェッチ対象ループ集合};
i=1;
Q=L0の要素数;
904
i≧Q?
905
YES
NO
l0=L0のi番目のループ;
l0'=L0の(i mod Q)+1番目のループ;
l0.0=l0をインデックス集合分割した前半ループ;
l0.1=l0をインデックス集合分割した後半ループ;
α=主記憶からキャッシュへのプリフェッチサイクル/1回あたりの予測ループ実行サイクル;
l0のプリフェッチ対象メモリ参照X[i]に対して、l0.0にX[i+Step(l0)*α]に対するプリフェッチ命令を挿入;
l0'のプリフェッチ対象メモリ参照Y[j]に対して、βをl0.1開始時のループインデックスの値、
Dをl0とl0'のループ増分値の比とすると、l0.1にY[Start(Y)+(k-β)D]に対するプリフェッチ命令を挿入;
i=i+1;
906
END
907

【図１３】

図 13

```
DO J=1,M
  DO I=1,N
    X[J][I] = Y[J][I]
  END DO
END DO
```
201

(a)オリジナルループ

```
DO J=1,M
  DO I=1,N
    PREFETCH X[J][I+α]
    PREFETCH Y[J][I+α]
    X[J][I] = Y[J][I]
  END DO
END DO
```
202

(b)プリフェッチ挿入後のループ

```
DO J=1,M
  DO I=1,α
    PREFETCH X[J][I]
    PREFETCH Y[J][I]
  END DO                          } 203
  DO I=1,N-α
    PREFETCH X[J][I+α]
    PREFETCH Y[J][I+α]
    X[J][I] = Y[J][I]             } 204
  END DO
  DO I=N-α+1,N                    } 205
    X[J][I] = Y[J][I]
  END DO
END DO
```

(c)全要素に対するプリフェッチ
を行なう場合のループ